

Creating and Modifying a Vector Graphics Method

Â

[Contents](#) [Previous](#) [Next](#)

Goal: Guide you through creating and setting vector graphics method attributes.

Before running the tutorial below, type "python" or "cdat" at the command line.Â You will see the python prompt appear (i.e., ">>>"). You can now enter the command lines below.

You can [view](#)Â or [download](#)Â the full source code. To run the source code at the command line, type: "python vector_file.py".

```
# Import the modules needed for the tutorial
# cdms - Climate Data Management system accesses gridded data.
# vcs - Visualization and control System 1D and 2D plotting routines.
# cdutil - Climate utilizes that contains miscellaneous routines for
#           manipulating variables.
# time - This module provides various functions to mainpulate time values.
# os - Operation System routines for Mac, DOS, NT, or Posix depending on
#      the system you're on.
# sys - This module provides access to some objects used or maintained by
#       the interpreter and to functions that interact strongly with the interpreter.
import vcs, cdms, cdutil, time, os, sys
```

```
# Open data file:
filepath = os.path.join(sys.prefix, 'sample_data/clt.nc')
cdmsfile = cdms.open( filepath )
```

```
# Extract two 3 dimensional data sets and get a subset of the time dimension
data1 = cdmsfile('u', longitude=(-180, -48.75), latitude = (10., 70.43))
data2 = cdmsfile('v', longitude=(-180, -48.75), latitude = (10., 70.43))
```

```
# Initial VCS:
v = vcs.init()
```

```
# Show the list of persistent vector graphics methods.
v.show('vector')
```

```
*****Vector Names List*****
(  1):          ASD_map          default          polar
(  4):          quick           robinson
*****End Vector Names List*****
```

Get a vector graphics method object and plot:

```
# Assign the variable "vf_asd" to the persistent 'quick' vector graphics methods.
vf_asd = v.getvector( 'quick' )
```

```
# Plot the data using the above vector graphics method.
v.plot( data1, data2, vf_asd )
```

```
# List the 'quick' vector graphics methods attributes.
vf_asd.list()
```

```
-----Vector (Gv) member (attribute) listings -----
```

```

Canvas Mode = 1
graphics method = Gv
name = quick
projection = linear
xticlabels1 = *
xticlabels2 =
xmtics1 =
xmtics2 =
yticlabels1 = *
yticlabels2 =
ymtics1 =
ymtics2 =
datawc_x1 = 1.00000002004e+20
datawc_y1 = 1.00000002004e+20
datawc_x2 = 1.00000002004e+20
datawc_y2 = 1.00000002004e+20
datawc_timeunits = days since 2000
datawc_calendar = 135441
xaxisconvert = linear
yaxisconvert = linear
line = None
linecolor = None
linewidth = None
scale = 1.0
alignment = center
type = arrows
reference = 1.00000002004e+20

# Change the vector scale value
vf_asd.scale = 5.0

# Change the vector scale value to a negative number
vf_asd.scale = -5.0

# Change the vector type to wind barbs
vf_asd.scale = 2.0
vf_asd.type = 1

# Change the vector type to solid arrow head
vf_asd.scale = 3.0
vf_asd.type = 2

# Change the vector reference and vector alignment
vf_asd.scale = 1.0
vf_asd.type = 0
vf_asd.reference = 15.
vf_asd.alignment = 'tail'
vf_asd.linecolor = 242

```

```
# Create a persistent vector graphics methods from an existing vector graphics method.
vf_new = v.createvector( 'new', 'quick' ) # create new from quick
vf_new2 = v.createvector( 'new2','quick' )# create new2 from quick
vf_new.list()                            # list its attributes
v.show('vector')                          # show vector list with new and new2
v.removeobject( vf_new )                  # remove new from vector list
v.show('vector')                          # show vector list without new
```

Â

[Contents](#) [Previous](#) [Next](#)